University of Waterloo
Faculty of Engineering
Department of Electrical and Computer Engineering

Motion Capture Lite

Final Report

Group Number 14
Prepared by:

Name	Student Number	Email
Aryaman Singh	20815224	a55sing@uwaterloo.ca
Arjun Mehta	20839709	a47mehta@uwaterloo.ca
Kevin Kalathil	20846153	kkalathi@uwaterloo.ca
Nicolas Bao	20828231	n3bao@uwaterloo.ca

Consultant Name: Murray Dunne

Date: February 25, 2024

Abstract

Motion Capture is a common technique used to track the movement of objects digitally. Used extensively by the entertainment, sports, and robotics industries, existing solutions are generally expensive and complex. They typically involve elaborate suits, large camera setups, and human-centered tracking. The objective of this project is to build a “lite” motion-capture device that can accurately track small objects that are normally too difficult to track using just a camera. The device uses semi-passive imperceptible markers, which refer to projectors that illuminate small LED marker tags placed on objects. The marker tags are equipped with photosensors to detect light from a projector, and a microcontroller. The markers will process and communicate the results back to a computer where a software tool, through signal processing, converts recorded motion data from the device into 3D motion vectors and allows for real-time tracking. This method allows the device to capture the motion of objects at high speed, using relatively simple and cheap electronics (photosensors and LEDs).

Acknowledgments

We want to, first and foremost, express our gratitude to Professor Murray Dunne for his invaluable assistance over the past year. His guidance has played a crucial role in shaping the direction of this project since its inception, and we deeply appreciate his encouraging words and unwavering support throughout the entire process.

We are also extremely thankful to the original authors of *Prakash: Lighting Aware Motion Capture using Photosensing Markers and Multiplexed Illuminators [1]*, which was published by the Mitsubishi Electric Research Laboratories (MERL) in Cambridge, MA - this paper was the original inspiration for our work, and though our project differs in implementation, it has been essential in our work.

Finally, we are thankful to the entire ECE department, professors, and TAs for their continued support through the FYDP process.

University of Waterloo Territorial Acknowledgment

The University of Waterloo acknowledges that much of our work takes place on the traditional territory of the Neutral, Anishinaabeg, and Haudenosaunee peoples. Our main campus is situated on the Haldimand Tract, the land granted to the Six Nations that includes six miles on each side of the Grand River. Our active work toward reconciliation takes place across our campuses through research, learning, teaching, and community building, and is co-ordinated within the Office of Indigenous Relations.

Table of Contents

List of Figures

List of Tables

1. High-Level Description of Project

1.1 Motivation

Motion Capture is a common technique used to track the movement of objects digitally. The global market for motion capture is valued at around US$180M as of 2022 and is expected to reach US$600M by 2030. [2] Used extensively by the entertainment, sports, and robotics industries, existing solutions are generally expensive and complex, with most options costing thousands of dollars. [3] They typically involve elaborate suits, large camera setups, and human-centered tracking. The objective of this project is to build a “lite” motion-capture device that can accurately track small objects such as in-game props that are normally too difficult and too costly to track using a camera.

The device uses semi-passive imperceptible markers [1], which refer to projectors that illuminate small LED marker tags placed on objects. The marker tags are equipped with photosensors to detect light from a projector, and a microcontroller. The markers process and communicate the results back to a computer where a software tool, through signal processing, converts recorded motion data from the device into 3D motion vectors and allows for real-time tracking. This method allows the device to capture the motion of objects at high speed, using relatively simple and cheap electronics (photosensors and LEDs).

1.2 Project Objective

The objective of this project is to design a relatively low-cost motion capture system to track objects and props using semi-passive imperceptible markers and photosensors. The sensors process and send results to a computer, where the captured data can be used to visualize objects in motion.

1.3 Block Diagram

The block diagram for our project is as follows:

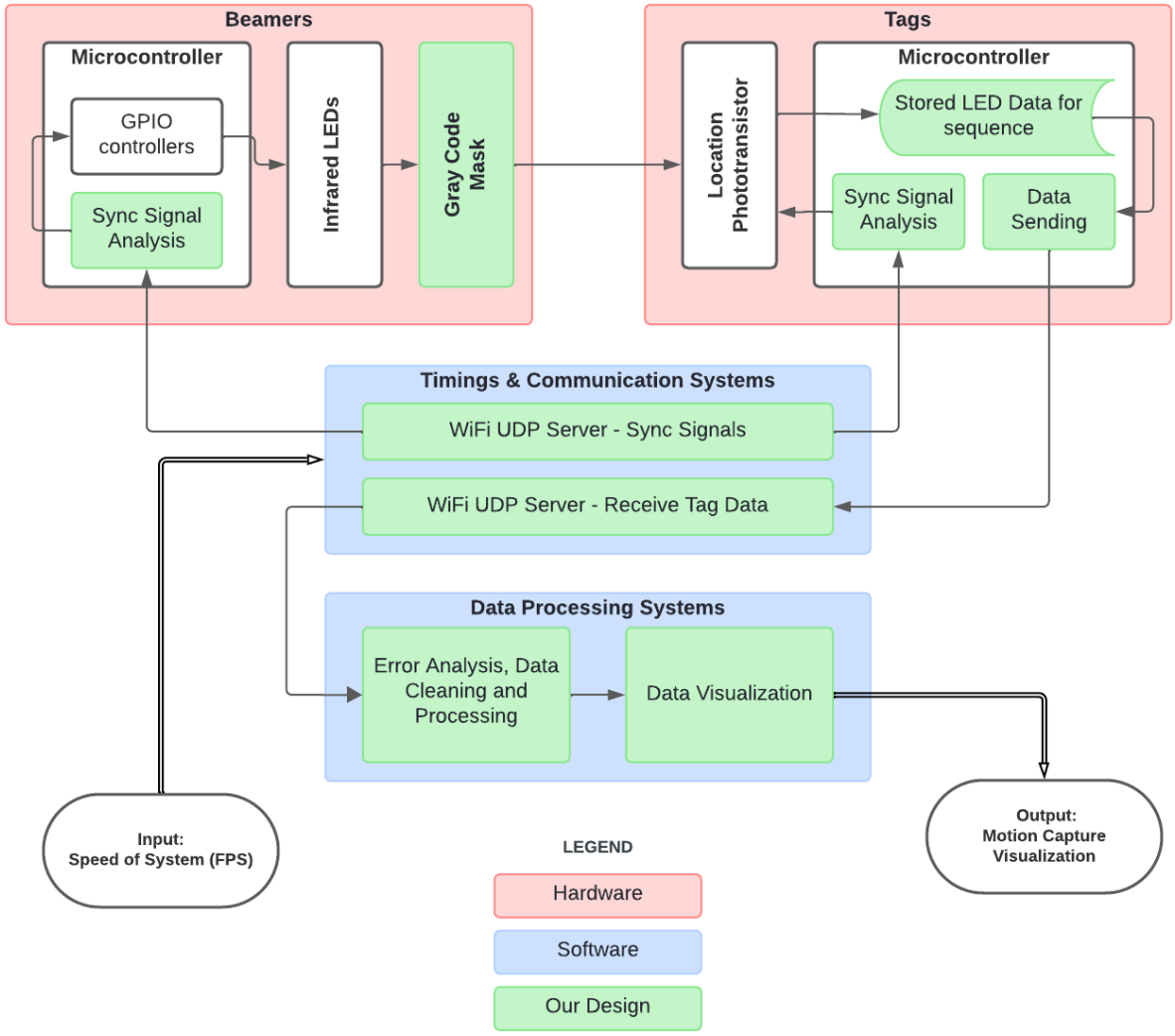

Figure 1. Block Diagram of Motion Capture Lite

We have 4 subsystems as part of our project:

1. **Beamers (Transmitters):** The beamer components consist of a microcontroller, which receives a SYNC signal to begin transmitting. The transmission happens using the GPIO pins of the microcontroller, which turn on each of the 6 LEDs sequentially, at speeds up to 10ms per LED. A gray code mask is placed in front of the LEDs to uniquely identify each beamer. A single beamer is used to identify a tag's location within a specific axis.

2. **Tags (Receiver):** The tags are to be attached to the object of interest and each tag will be used as a reference to describe the object's movement. The tag begins observing the LEDs using its phototransistor when it receives a SYNC signal, stores the sequential data for all 6 LEDs, and sends the data to the server when it has received all the required data.

3. **Timings & Communication Systems:** This subsystem consists of a server that sends a SYNC signal to all the beamers and tags at the same time, ensuring that the beamers and tags are coordinated correctly. This subsystem also receives the data from the tag microcontroller and sends it to the data processing systems.

4. **Data Processing System:** This subsystem takes in the voltage values communicated by the tag to the server, performs error correction and data analysis on it, and then parses the data to determine the position of the tag in 2D/3D space. Data visualization software is used to display the position.

2. Project Specifications

2.1 Functional Specifications

Specification	Subsystem	Description	Necessity
Sensor integration	Tags (Receiver)	Each tag contains a phototransistor which is used for location tracking.	Essential
Data Processing	Data Processing Software, Tags/Receiver	Take a bitstream from each tag and use Location Vector algorithms such as triangulation to convert raw data from the photosensors into 3D motion data	Essential
Communication	Beamer (Transmitter), Tags (Receiver)	A tag's microcontroller communicates results from phototransistors to the server. The computer sends a sync signal to all tags and beamers.	Essential
Data Storage	Tags (Receiver)	Tags should maintain a bit stream that will be used to resolve its position. This data is to be stored on the microcontroller's on-device Flash memory.	Essential
Real-time streaming	Data Processing Software, Tags (Receiver)	The data recorded by the microcontroller should be relayed to software that converts and processes the raw data stream. Real-time streaming enables a better user experience and decreased memory usage on the device.	Non-Essential

Table 1. Functional Specifications

2.2. Non-functional Specifications

Specification	Subsystem	Description	Necessity
Lightweight	Tags	As these tags are going to be placed on moving objects, they need to be light.	Essential
Cheap	Tags	Since we may need numerous tags on a single object or we want to track multiple objects simultaneously. We would like the tags to be cheap to produce.	Non-Essential
Portable	Tags	One of the main benefits of this type of motion capture is the simplicity of the system components. This simplicity can also allow for more portable devices	Non-Essential
Scalable	Software, Tags	The system should scale with any number of sensors that are added to the system.	Non-Essential

Table 2. Non-functional specifications

3. Detailed Design

3.1 Tags

3.1.1. Physical Design

The design of the tag is relatively simple - it consists of a phototransistor that is designed to be able to detect when the infrared LED is turned on. The requirements of the phototransistor are as follows:

Angle of Detection	**Intensity falloff vs distance**	**Intensity falloff vs angle**	**Light spectrum**
Narrow	Minimum	Sharp	Infrared

Table 3. Requirements for phototransistor on tag

The phototransistor that we selected is the *Würth Elecktronik 1541201NEA400 [4],* which fulfills all of these requirements. It is important to note that in the context of our circuit, the phototransistor acts as a current source, and the ADC of the microcontroller reads the voltage across the resistor. This can be seen in the figure below:

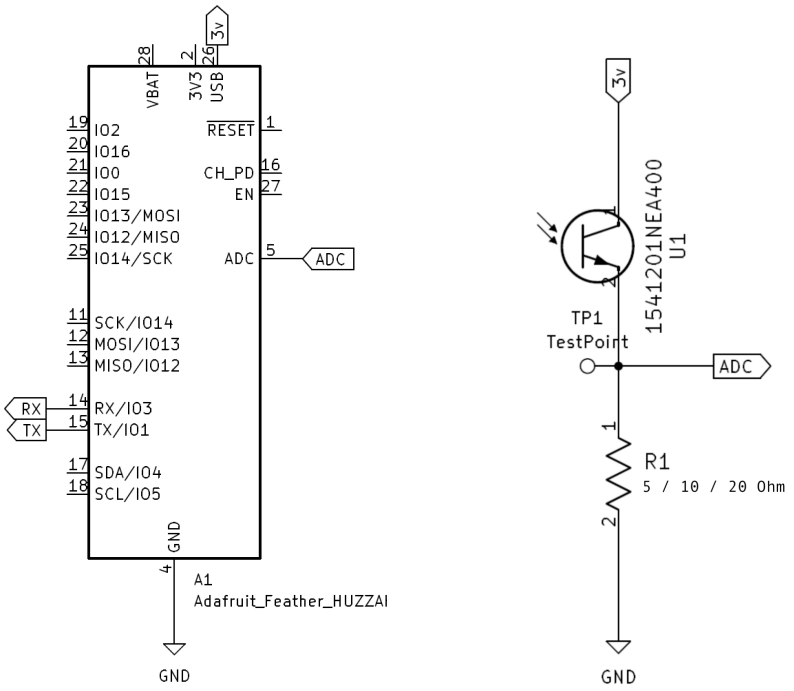

Figure 2/3. Diagrams showing connections on the tag microcontroller (left) and the phototransistor circuit (right). Note the R1 resistance on the phototransistor circuit.

Thus, our phototransistor can have different sensitivities depending on the value of the resistance. Different sensitivities have different relationships with the voltage values being read by the ADC (Analog to Digital Converter) in the tag. To quantify this relationship, we placed an infrared LED a certain distance away from the phototransistor and observed how the ADC values changed depending on the sensitivity of the photosensor:

Sensitivity of Photosensor Resistor (R1)	**Distance between phototransistor (tag) and LED (beamer)**			
	1- 2 inches[1] (High Value)	2 feet	4 feet	LED Off (Base Value)
Low - 5 Ohm	27	14	7	4
Medium - 10 Ohm	120	64	12	7
High - 20 Ohm	673	104	31	8

Table 4. Variation of ADC Values from the microcontroller depending on the distance between the tag and beamer (i.e. phototransistor and IR LED)

[1] This distance is impractical for the final product due to the 3D printed casing around the tags/beamers, but gives us an idea of the highest possible value the phototransistor can output.

As anticipated, the highest sensitivity resistor yielded the widest range of ADC values. Despite being conducted in ideal conditions, this extended range suggests reliable LED differentiation even in unfavorable lighting. While a higher sensitivity may result in a slightly elevated base value (only 1-4 units more than lower/medium sensitivity resistors in our tests), the benefits outweigh this minor drawback.

Below, we can see our final tag PCB along with the microcontroller:

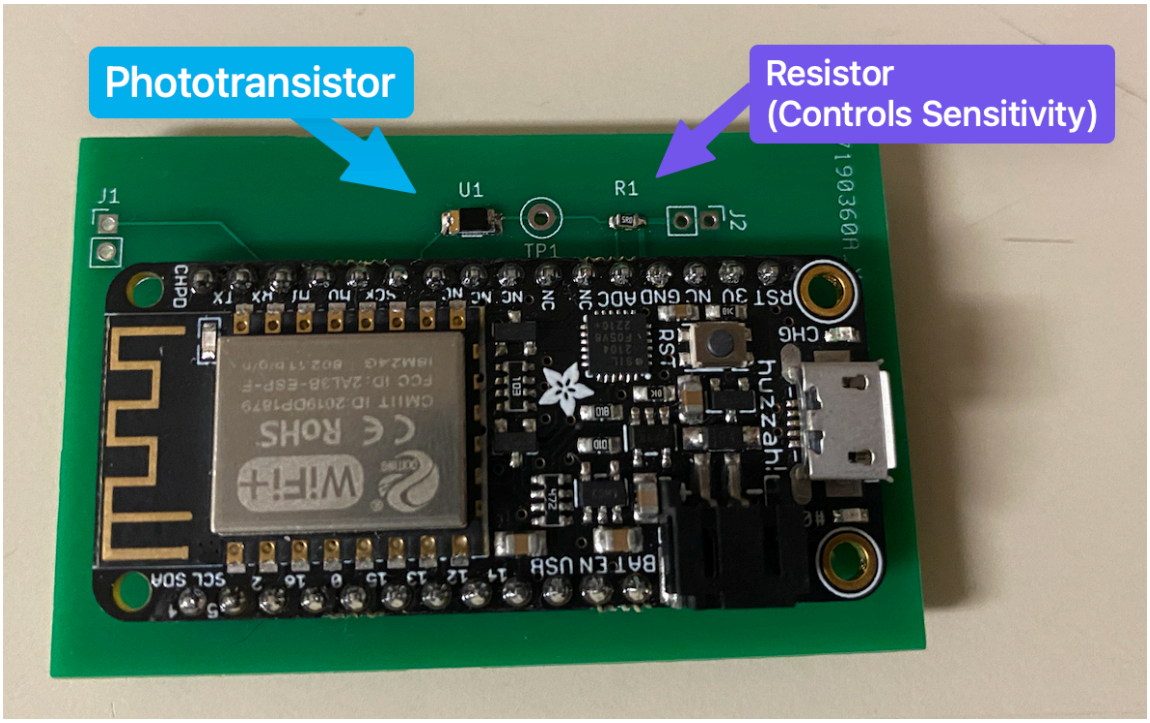

Figure 4. Image displaying PCB for tag and microcontroller

3.1.2 Software Design

The tags are required to meet certain timing deadlines which makes them real-time systems. The software for the tag is responsible for detecting the gray-coded LED signals from the beamers that are being polled by a controller. The tag begins detecting light from the LEDs when the synchronization signal is received from the WiFi server. It periodically checks if it detects an LED, and each time period corresponds to a different LED. At any given point, only one LED would be on. Originally, we had designed our tags to decode the gray-coded signals themselves, by setting a minimum and maximum voltage reading which would count as an LED being on. However, we

now send the raw ADC values for each sequence directly to the WiFi server, thus reducing the processing on the tag end.

The overall algorithm can be simplified to these 4 steps:

1. **Receive a SYNC signal from the server:** Note that since reading the contents of a UDP packet takes time, we simply use the presence of a UDP packet to indicate synchronization.
2. **Set inputVoltage for LED:** During the reading window for any given LED, set inputVoltage to be the maximum voltage read in that window
    - *Note that voltage here does not refer to V, but rather the unitless value generated by the ADC.*
3. **Add current value to packet:** When the reading window for any given LED is over, add the inputVoltage to the packet.
4. **Send packet to server:** When all 6 LEDs have been read, send the packet to the server.

```
int value = -1;
void loop() {
  currentTime = millis();
  int packetSize = Udp.parsePacket();

  if (packetSize) {  // 1. Received Sync Signal
    value = 0;
    startReading = true;
    startTime = currentTime;
    delayStartTime = startTime + timeDelay;
    delayEndTime = startTime + timeWindow - timeDelay;
  }
  if(startReading) {
    if(currentTime >= delayStartTime && currentTime <= delayEndTime) {
      inputVoltage = max(inputVoltage, analogRead(ANALOG_PIN)); // 2. Set inputVoltage to maximum observered voltage
    }
    if (currentTime - startTime >= timeWindow) {
      startTime = startTime + timeWindow;
      delayStartTime = startTime + timeDelay;
      delayEndTime = startTime + timeWindow - timeDelay;
      if(value == 5) {
        sprintf(currentPacket, "%d:%d", value, inputVoltage);
      } else {
        sprintf(currentPacket, "%d:%d,", value, inputVoltage); // 3. Add the current value to the packet
      }
      strcat(replyPacket, currentPacket);
      inputVoltage = 0;
      if(value == NUM_LEDS - 1) {
        startReading = false;
        Udp.beginPacket(Udp.remoteIP(), portToSend); // 4. Send the packet to the server
        Udp.write(replyPacket);
        Udp.endPacket();
        memset(replyPacket, 0, sizeof(replyPacket));
      } else {
        value += 1;
      }
    }
  }
}
```

Figure 5. Arduino (.ino) code snippet for tag microcontroller, showing the four steps for reading, storing, and sending the values.

3.2 Beamers

3.2.1 Physical Design

The beamer's core function is to control the on/off states of 6 location LEDs. These LEDs, along with the gray code mask, enable the tag's positional detection of the beamers. The pivotal element in the beamer design is the LED, which must meet the following requirements:

Brightness	**Cost**	**Angle**	**Wavelength**
High	Low	Narrow (can be done with physical design)	Infrared

Table 5. Requirements of Location LED on Beamer

The LED we chose is the *VSMA 1094250 [5],* which fulfilled all of these requirements. Only 1 LED would be turned on at a time, which meant that we did not have to account for the power requirements of all LEDs being on at the same time.

However, since these LEDs require 3V at 1A (which is more than the board power supply of 3.3V at 250 mA), an external power source is needed. A 9V battery is chosen as our power source due to its simplicity and high availability, and a fuse is attached to the battery for safety. A buck converter is added to step down the voltage to 3.3V, which can be seen in the diagram below:

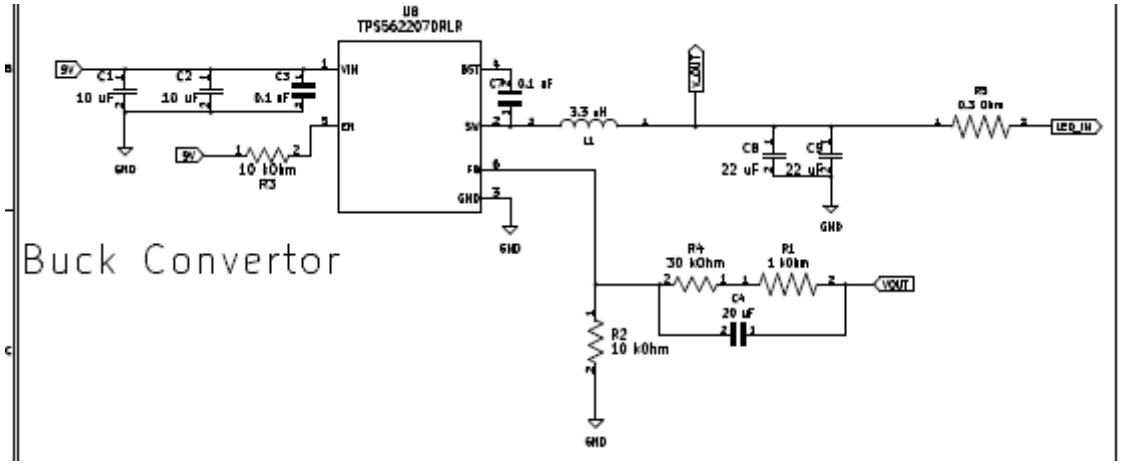

Figure 6. Circuit diagram for the buck converter on the beamer PCB

Mosfets in series with the LEDs allow us to control which LEDs are on via the GPIO pins of the microcontroller. Figures showing the configuration of the LEDs and Mosfets, as well as how the

GPIO pins control the Mosfets, can be seen below:

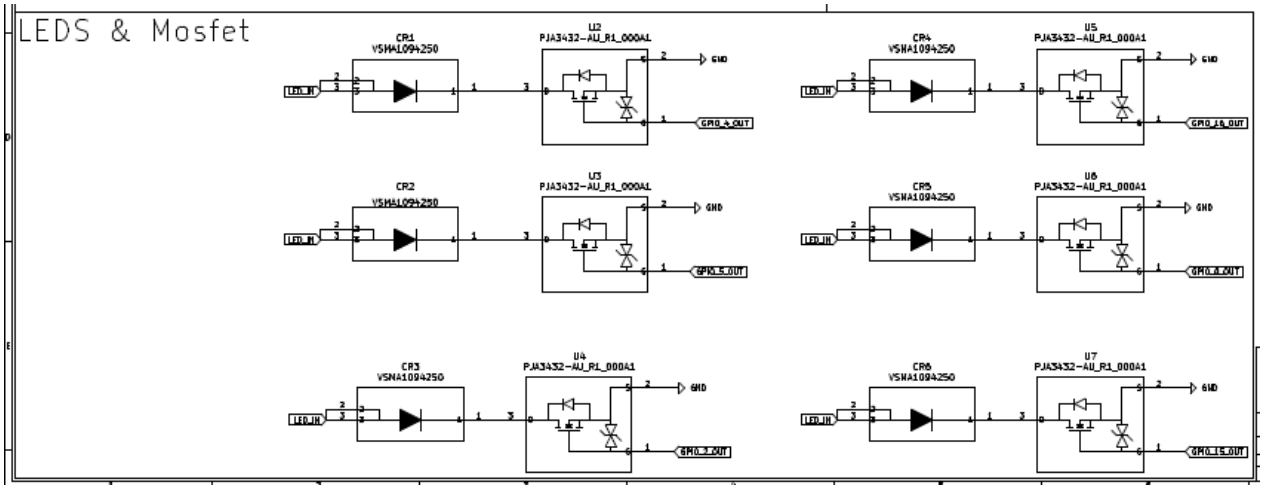

Figure 7. LED and Mosfet diagrams for beamer PCB

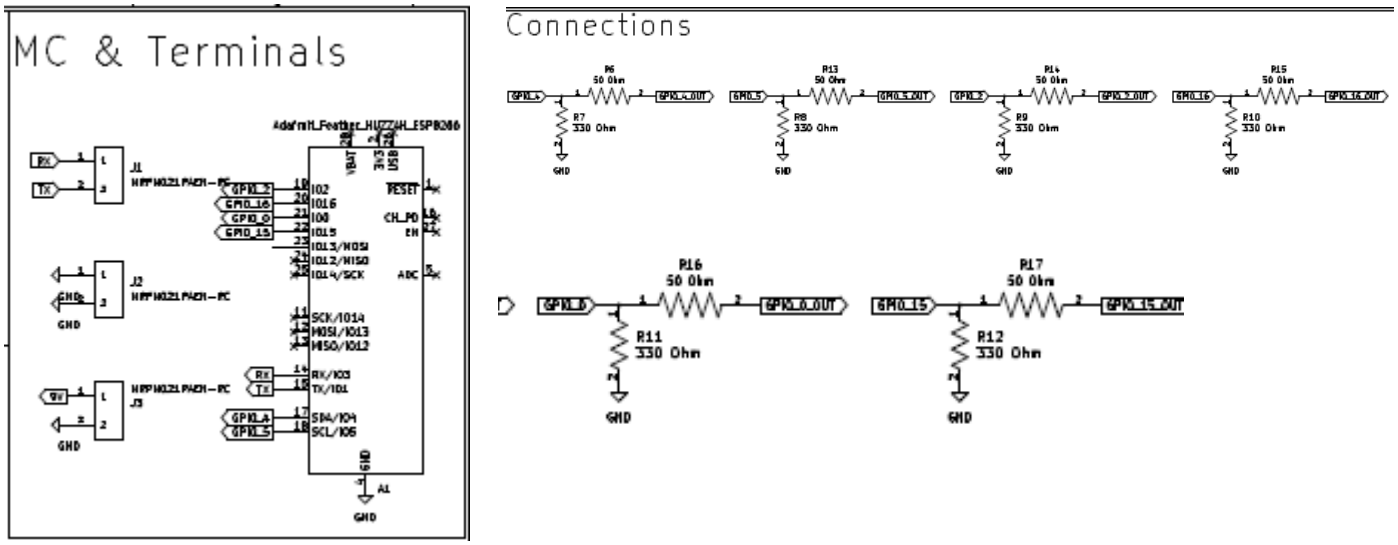

Figure 8. Circuity diagram showing MC terminal connections (left) and GPIO connections (top)

Another important aspect of the physical design of the beamers is the 3D-printed casing, which is done to protect the LEDs and other components on the PCB. The designs and final product can be seen below:

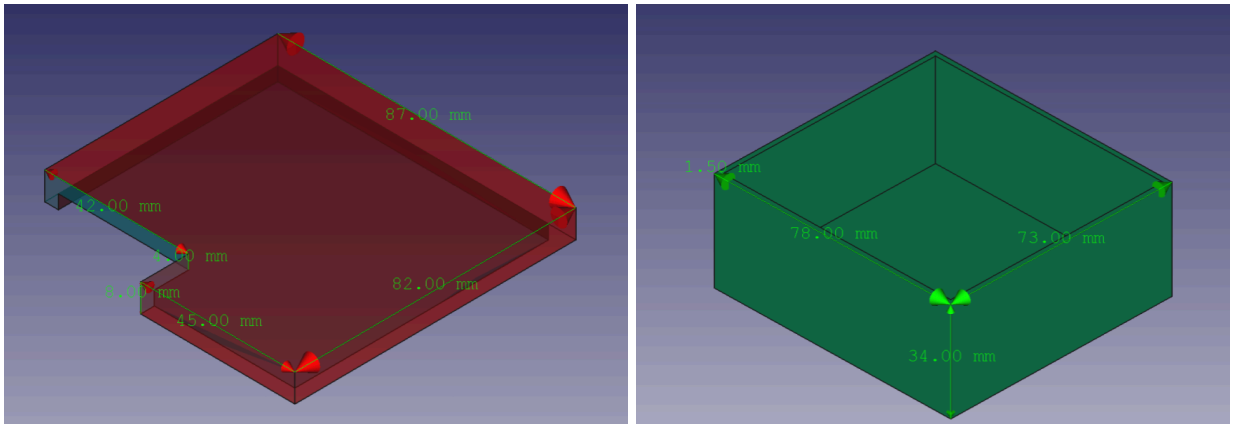

Figure 9. 3D models of the beamer casing

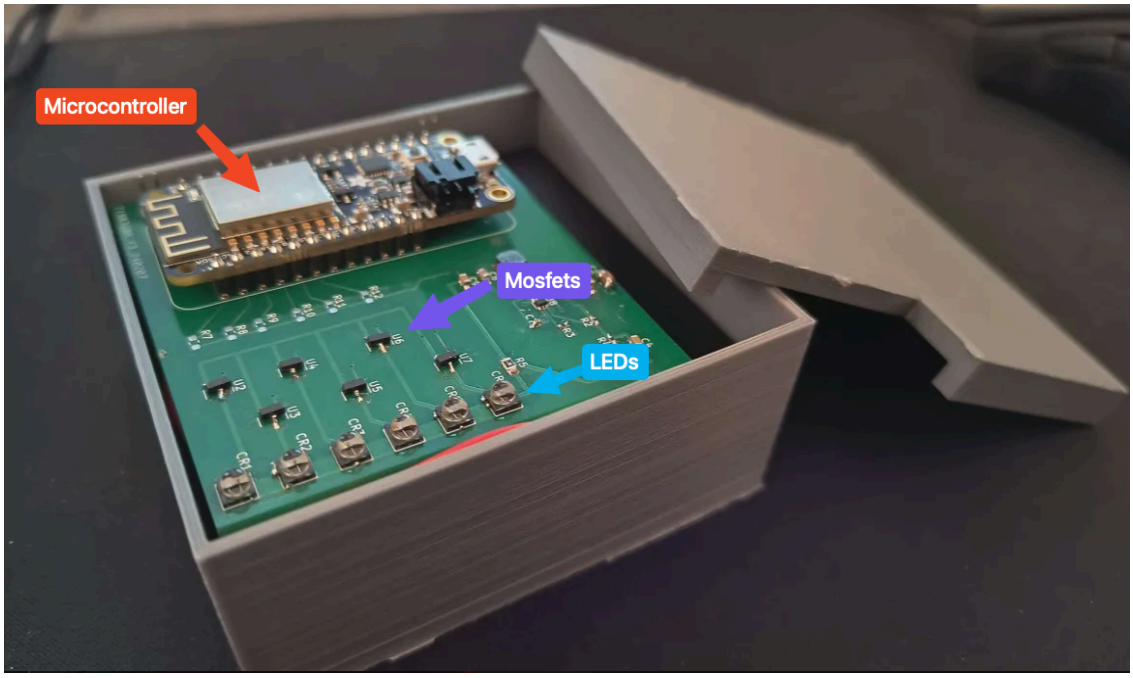

Figure 10. Sample 3D-printed casing with microcontroller and PCB - important components are labeled in the diagram for convenience.

3.2.2 Software Design

The beamers play a crucial role in the timing and speed of our entire system. The rate at which we can cycle between LEDs directly affects the overall speed (i.e. frames per second) we can get in our motion capture system. This requires us to determine the fastest rate at which the LEDs can be cycled through[2] while ensuring that the phototransistor accurately captures each LED without any miscount or mapping errors.

[2] For more information about the timing system, see Section 3.3 below.

To test this, we turn on only one LED out of the 6 (LED #3 in this case), and observe how fast we can go without the phototransistor *missing* the LED or thinking it came from another LED. In other words, the 6 LEDs output `0 0 1 0 0 0`, and we expect the phototransistor to return this value. For each time window, we ran this experiment for 100 cycles.

Time Window LED stays on	**Number of cycles with missed LEDs**	**Number of cycles with leaks into previous/next LED reading (i.e LED 2 or LED 4)**	**Comments**
100ms	0	1	The phototransistor can sense the LEDs correctly most of the time for the given time window
50ms	0	2	The phototransistor can sense the LEDs correctly most of the time for the given time window
10ms	5	8	The phototransistor can sense the LEDs correctly with a few misses.
6ms	30	50	The tag is unable to sense the LED within the given time window for a large number of cycles

Table 6. Performance of LED detection depending on reading window

For the 6ms time window, we observe a large number of leaks where the tag believes the phototransistor reading corresponds to the previous/next LED, which could be a result of a slight delay between the tag and beamer receiving the SYNC signal. This indicates that this time window is too small for the phototransistor to sense the LED reliably.

Thus, we concluded that we could go as fast as 10ms per LED without compromising the accuracy of the phototransistor readings (and hence the location calculation) for the tag. A smoothing algorithm and results from previous cycles can be used to make up for any missed readings.

3.3. Timings and Communication

3.3.1 Time Modulation Decision

One of the downsides of using a phototransistor is its limited resolution as they can not differentiate between light sources. The big question, then, is - how do we avoid detecting light from the wrong LED?

One solution would be to use LEDs that emit light at a specific wavelength and have only one type of phototransistor that can detect that wavelength. There are many issues with this approach, mainly that getting LEDS and phototransistors that can emit and detect in a narrow band of the spectrum is expensive and not aligned with our goal of making the final model product affordable.

We are limited by the cost of LEDs and transistors, so our other option is **time modulation**. By only reading the values from our sensors during specific time intervals we can “allocate” a time for our beamers to communicate with our sensors.

3.3.2 Beamer Timing Design

The beamer’s sole function is to display a gray code light pattern for tags to observe.

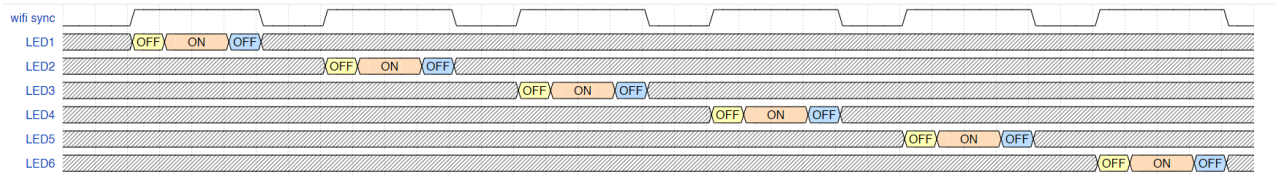

Figure 11: Timing diagram for one cycle of the beamer’s six LEDs

To further elaborate on how the gray code is sent, the LEDs are lined up and turned on and off in a sequence. Each LED is covered by a 3D-printed gray code mask which blocks light within certain regions only allowing tags within certain regions along the beamer’s plane to observe the emitted light. Since they are enabled sequentially, with each result (observed or not observed), we can halve the possible region where the tag may be located. Refer to Figure 4 for the general layout and timing of a 6 LED tag and gray coding.

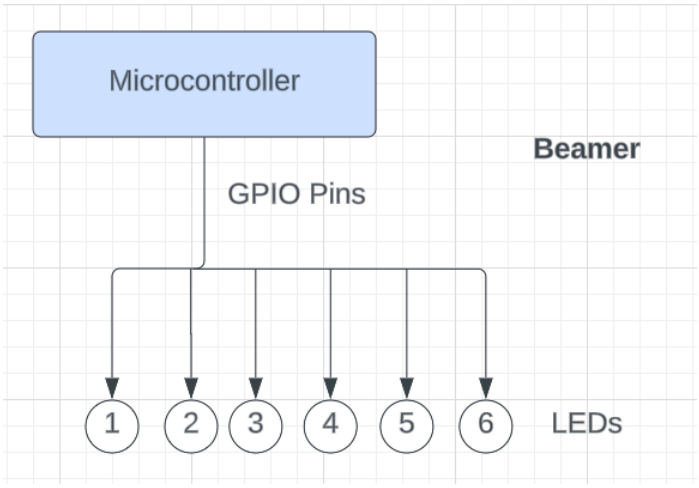

Figure 12: Block diagram of a 6 LED beamer

3.3.3 Communication Protocol - WiFi

Since the system requires high-speed transmission and low-latency data streaming, communication between server-tag and server-beamer is done wirelessly using the UDP protocol, and we use our own router to establish a stable wireless connection between the devices. The UDP socket is set to non-blocking mode, allowing continuity in the case of lost packets. We can handle the lost data by using data from the previous cycle and integrating smoothing algorithms.

The server sends a SYNC signal to a single beamer as well as all the tags at the same time, each listening on their own IP address respectively. Upon receiving the SYNC signal, the beamer turns on its LEDs one-by-one in sequence. The tags return a sequence containing the maximum photo sensor reading they observed for each LED to the server (which is always listening on its IP address on a separate thread). The server then repeats the cycle for the next beamer.

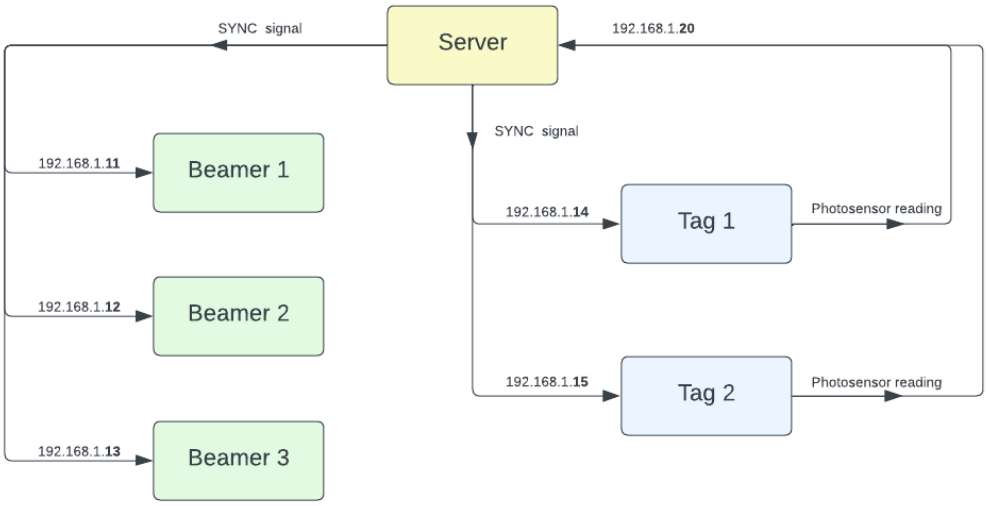

Figure 13: Block diagram depicting wireless communication between server-beamer and server-tag

Note that the IP addresses used in Figure 6 are placeholders, and they can be configured as required.

So far we have looked at timings for one tag and one beamer, but for our motion capture system to work in 3 dimensions, we need at least three beamers. If the beamers are independent of each other and fire off their LEDs at random, we encounter the same issue of not being able to identify the origins of the LEDs. To resolve this problem, we have decided to synchronize the beamers and have them fire in sequence after one another. To do this, we have to establish a way to coordinate communication across separate beamers.

To manage this task of coordinating separate beamers and tags, we synchronize the entire system using a centralized server stub that handles wireless data transmission, receiving, and processing.

The server, which is a program running on a separate computer, sends out periodic sync signals to all devices (beamers and tags) such that they all know when to emit or observe light, depending on whether the device is a tag or beamer. With the target of 5 frames per second (FPS), we would need to hit an effective clock speed of approximately 11ms per beamer.

$$\frac{1000\ \text{ms}}{5\ \text{fps}} = 200\ \text{ms per frame}$$

$$\frac{200\ \text{ms per frame}}{18\ \text{beamer cycles}} \approx 11\ \text{ms per beamer cycle}$$

We originally intended to send a separate sync signal for each packet but quickly realized that the low latency and high number of packets being transmitted was not feasible.

The following are the results of a test we performed to determine how quickly we could transmit data from the server to a single microcontroller. On the server code, we transmitted a counter value and on the receiver end, we logged the received counter value. If a counter value is missed, that is considered a drop. We tested by sending counter values from 1-200 and analyzing the output on the microcontroller logs.

Time Interval between synchronization signals	**Number of dropped packets**	**Comments**
50ms	0	The target device receives each packet in order and on time
10ms	<=2	Based on our trials, we never drop more than 5 packets at 10 ms which is recoverable with smoothing, but still not ideal
5ms	<=20	20 missed packets in 200 transmissions is not ideal/recoverable.

Table 7. Variation of dropped packets with respect to interval between sync signals

Based on these test results, we then opted for a batching/sequencing approach where the sync signal received represents six cycles. In the case of the beamer, it executes its complete six LED sequence after receiving a sync signal, and the timing delays are handled within the microcontroller code that is flashed on the board. This increases the time between sync signals and decreases the number of packets transmitted by a factor of six. Hence, to achieve an effective clock cycle of 10ms for each LED being toggled, we only need to send packets every 60 ms which is achievable given the current system and its limits (as discussed in Section 3.2.2).

3.4 Data Processing Software

3.4.1 Software Architecture

Before beginning programming the data processing software, we first conceived what the design architecture should look like. As per our ECE 452 knowledge, we decided that a *Pipe & Filter* architecture would fit our project the best. Pipe & Filter architecture specializes in software that is heavy in processing, and where the flow of data is fairly linear. See Figure 7 to see an example of Pipe & Filter architecture used to determine the location of the tags using sensor data:

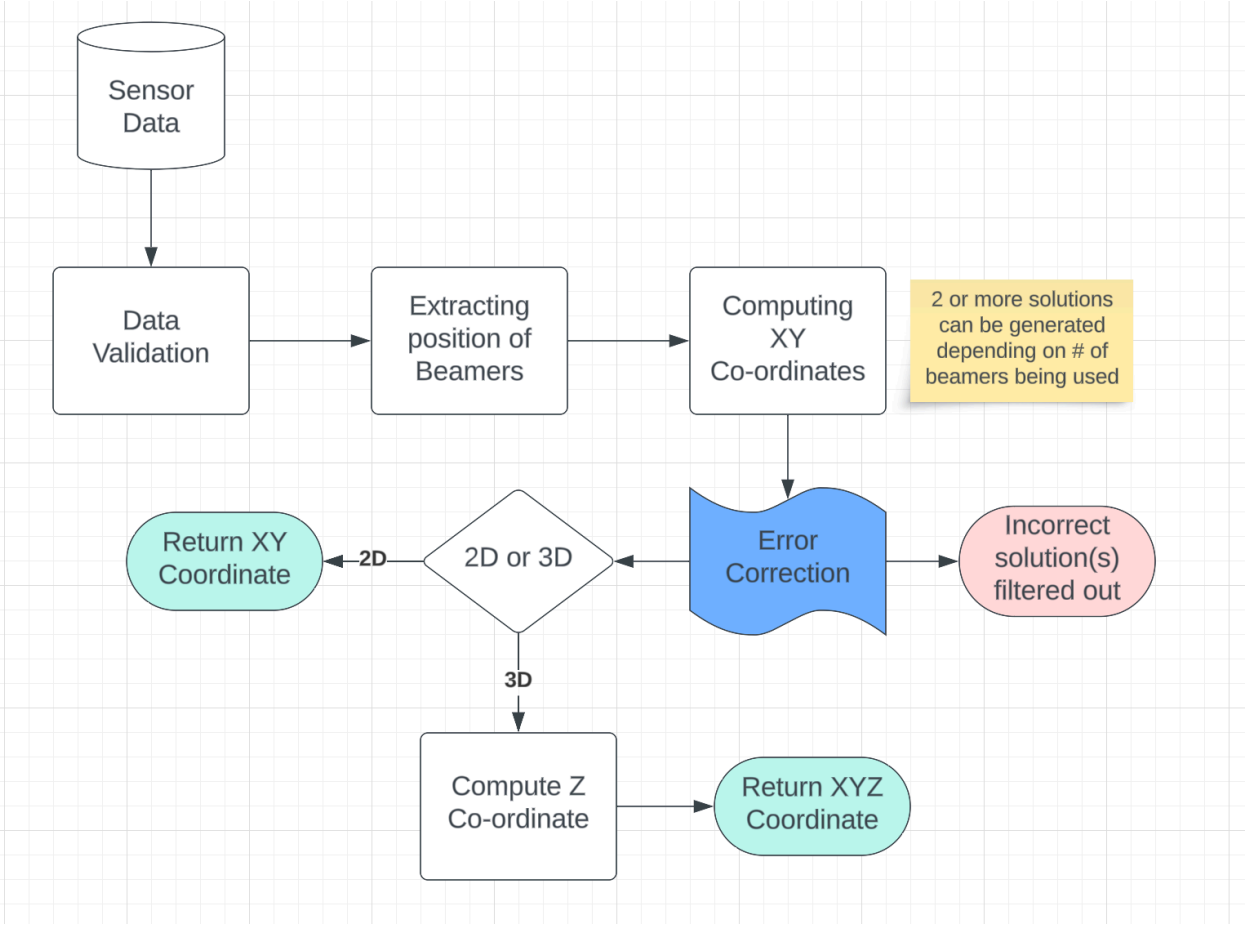

Figure 14. Flow Architecture of the Location Processing Algorithm. Pipe & Filter architecture can be clearly seen by the each architectural component is a process, and the flow of the data is linear. Error Connection symbolizes one of the filter components.

Data validation is done by simply ensuring that the values received from the tag match the range of the ADC values of the phototransistor, which is determined during our testing of the tag (as explained in Section 3.1.1).

3.4.2 Data Visualization

Once we have the coordinates of the tags, our next goal is to visualize the data. Our primary emphasis in developing the motion data visualization is to ensure that we design a system that is fast, easy to run on low-scale hardware, and easy to implement. Given the importance placed on our hardware within our project scope, our objective is to design a system that seamlessly integrates with our hardware, minimizing potential complications.

During the consideration of data visualization software, we explored various options, such as Python's Matplotlib, and Rust's egui, among others. We decided to use a decision matrix to come to the final conclusion on which visualization software to use, with a scoring range from 1 (Low) to 3 (High):

Criteria	**Matplotlib (Python)**	**EGUI (Rust)**	**Qt (C++)**	**Tkinter (Python)**	**Electron (JS/TS)**
Ease of Integration[3]	High (3)	High (3)	Low (1)	High (3)	Low (1)
Performance (higher FPS, animation smoothing)	Moderate (2)	High (3)	High (3)	Low (1)	Low (1)
Cross-Platform Support	High (3)	Moderate (2)	Moderate (2)	Medium (3)	High (3)
Flexibility (ability to create custom GUI components)	Moderate (2)	Low (1)	Moderate (2)	Moderate (2)	High (3)
Prior Team Experience	High (3)	Moderate (2)	Low (1)	Medium (2)	High (3)
Visual Appeal	Moderate (2)	Low (1)	Low (1)	Moderate (2)	High (3)
Total	15	12	10	13	14

Table 8. Decision Matrix for visualization tool

[3] During the discussion on which visualization software to use, we were already using *Rust* and *Python* for parts of our server and networking code, which is why tools that used these languages were given a *High* rating.

In the end, we ended up using *Python's Matplotlib,* primarily due to its ease of use and easy integration with our existing codebase. Another advantage of *Matplotlib* is its easy implementation of 3D visualization [6], which some other visualization tools did not support.

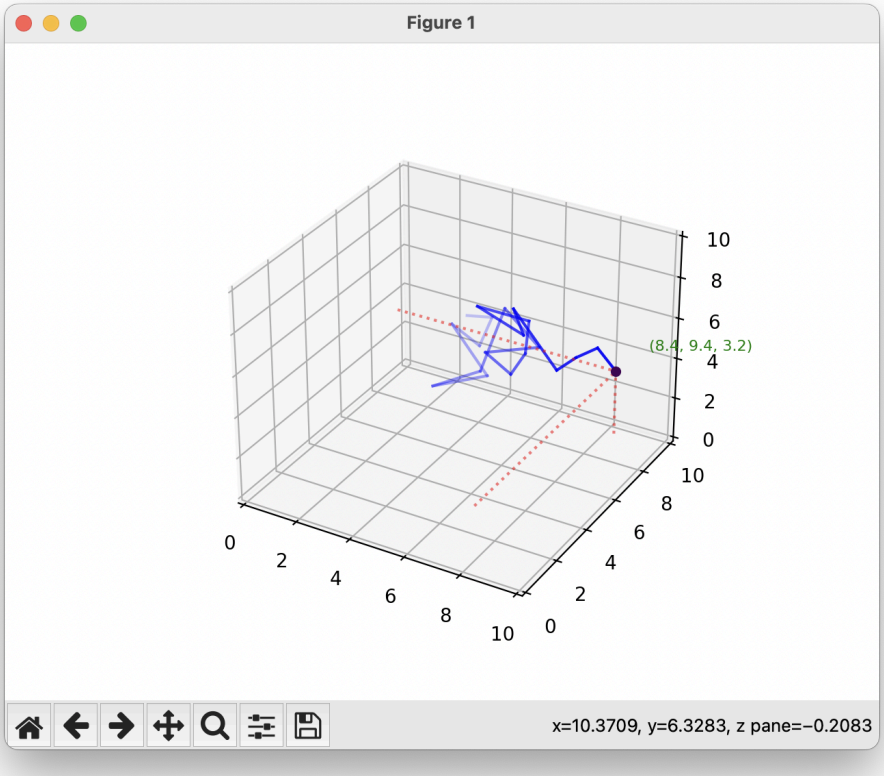

Figure 15. Demo of Data Visualization using Python's Matplotlib. The point is labeled and has red lines indicating shadow on the axes.

A very simple animation algorithm is applied to ensure a "smooth" path of the motion capture data can be shown. The previous 20 (or so) points were stored and graphed, and lines connecting these points with decreasing transparency were added to visualize a "trail" behind the point:

```
# Initialize a list to store the last 20 points
prev_points = [[0, 0, 0]]

def update(frame):
    global prev_points

    ax.clear()
    ax.set_xlim([0, 10])
    ax.set_ylim([0, 10])
    ax.set_zlim([0, 10])

    prev_points.append((x, y, z))

    if len(prev_points) > 20:
        prev_points.pop(0)

    # Draw lines between the points
    lines = []
    for i in range(len(prev_points) - 1):
        line, = ax.plot([prev_points[i][0], prev_points[i+1][0]],
                [prev_points[i][1], prev_points[i+1][1]],
                [prev_points[i][2], prev_points[i+1][2]], color='b', alpha=(0.2 + i/25)) # adjust alpha to fade lines
        lines.append(line)

    # Draw the actual point
    scatter = ax.scatter(x, y, z, c=z, cmap='viridis')
```

Figure 16. Showing how the trail animations were created in Python Matplotlib

Through this system, it is evident that we were able to achieve our requirements of real-time streaming of data from the tags to a visualization that can be seen, understood, and recorded with ease.

5. Discussion & Conclusions

5.1 Evaluation of Final Design

The objective of this project is to design a relatively low-cost motion capture system. The design proposed in the report meets the essential functional and non-functional specifications listed in Section 2.1 and Section 2.2 respectively. The design covers how the hardware and software of the tag (receiver) and beamer (transmitter) would be set up, as well as how they would communicate with the main software. The final design, integrated with 3D visualization, meets non-essential functional requirements such as real-time streaming, allowing users to track objects being moved in real-time.

Functional Requirement	**Status**	**Description**
Sensor integration	Done (tag)	Integration of phototransistor on tag is done successfully, as shown in section 3.1
Data Processing	Done	Data visualization is done successfully, as shown in section 3.4
Communication	Done (between tag, beamer and server)	Communication is achieved through WiFi and UDP, as shown in Section 3.3
Data Storage	Done (on tag itself)	Tags are successfully able to store the data for each LED locally, before sending it to the server.
Real-time streaming	Done (demo of visualization)	Data visualization can be done in real-time, as shown in Section 3.4

Table 9. Evaluation of functional requirement analysis

As for non-functional requirements, the hardware components comprising the tag were carefully picked to ensure that the tags were compact and lightweight. A 3D-printed container is also created for the devices to ensure safe, portable, and easy usage of the motion capture system.

5.2 Use of Advanced Knowledge

During the research, analysis, and design phases of the project, knowledge from upper-year ECE courses including ECE 455, 452, 458, and 327 were required.

Due to the need for controlling electronics on our projectors and tags, we needed to use our understanding of Embedded Software programming (**ECE 455 & ECE 327**) to identify microcontrollers that would most appropriately satisfy the timing requirements of the projector along with the form factor requirements for the tags. In order to meet the timing requirements of the projector, which iterates through the different gray codes through time modulation. When selecting the Huzzah ESP 8266 microcontroller, we factored in the clock speed which impacts the WCET of the system which was one of the major topics covered in **ECE 455: Embedded Software.**

In order to analyze the synchronization, timing, and communication of the various subsystems, knowledge of timing diagrams was required in order to outline expected workloads, which we had learned in **ECE 327: Digital Hardware Systems.** Additionally, we made sure to consider various timing delays including setup/hold times, and clock jitter which we accounted for in timing diagrams by including delays between state changes.

For the software sub-systems in our projects, we made heavy use of the software architecture skills we were taught in **ECE 452: Software Design & Architecture**, such as the pipe and filter system described earlier. We also used *Rust* and its many performance paradigms (such as Tokio and multithreading) while designing our server and networking systems, which we learned about in **ECE 459: Programming for Performance.** We also made heavy use of the networking concepts taught to us in **ECE 358: Computer Networks** such as gateways, subnet masks, IP address configuration, UDP protocol, and routing.

5.3 Creativity, Novelty, Elegance

One novel aspect of our project is that in contrast to conventional camera-based motion capture systems, where cameras capture the light reflected by the markers, our system utilizes projectors as the "cameras" that *transmit* the light, and tags on the tracked object to *receive* the light. This is the reverse of how the vast majority of motion capture (and camera systems) work in general. As discussed, this allows us to creatively use low-cost sensors and LEDs for the tracking.

A creative aspect of our project is the implementation of gray coding to identify different tags. Gray coding is a binary encoding scheme that minimizes errors when transitioning between marker states. This is most commonly used for digital signal communication in electronics and is not usually used in typical camera or computer-vision-based motion capture systems. Our implementation of this idea in the motion capture space is required because of our “inverted” motion capture, which requires us to use something like gray coding to uniquely identify the position of each LED tag.

Another novel aspect of our motion capture is imperceptible markers and photosensors. Unlike traditional systems that rely on active markers or sensors, we have eliminated the need for bulky tags that can interfere with the natural movement of the object being tracked.

5.4 Quality of Risk Assessment

The following table shows the risks we assessed during the 4A terms and the steps we took to address them.

Risk	**Description**	**Steps Taken**
Time management	Balancing time between FYDP and other demanding courses taken by the team members during the 4A and 4B academic terms	We set up weekly meetings, met in-person as much as possible, and recorded the project progress in a project management software, which helped us stay on track with tasks
Shipping delays in hardware components	Since these components are usually shipped from other countries, they can take long to be delivered	Being aware of this risk allowed us to order components ahead of time from reputable websites. We ordered the components we needed in the 4B term months before the term started
Slow software processing	The server (especially if written in Python) could struggle with processing large volumes of real-time data at high speeds, which is a requirement for our project	We migrated our server code from Python to Rust and integrated multithreading for sending and receiving data over WiFi to ensure our code is optimized to process data in real-time
Electrical circuitry safety risk	As the tags are supposed to be placed on an object, there could be a safety risk due to the electrical circuitry	We developed a 3D-printed enclosure for the tag and made sure to test only with dry objects, away from the vicinity of conductive liquids. Additionally, we use fairly large objects for testing to ensure that two tags on the same object are not in close proximity to each other. We also recorded chip temperatures during our testing to check for any thermal/fire hazards.

Table 10. Quality of Risk Assessment table

5.5 Student Workload

We met with the consultant every week to share our progress in the preceding week and to set goals for the upcoming week. We also conducted a team meeting every 2 weeks to discuss broader goals and prepare for upcoming deadlines. Additionally, we divided the work among team members and kept members working on hardware and software in sync through a group chat. The workload and contribution of each student are presented in the table below, based on the number of hours each student contributed towards the project and course deadlines.

Group Member	**Contribution/Workload (in %)**
Aryaman Singh	24
Arjun Mehta	22
Kevin Kalathil	24
Nicolas Bao	30

Table 11. Student Workload Table

The observed contribution/workload is close to the expected average workload of 25% per student. The slightly uneven workload distribution is due to one of the members being on co-op term during the summer (4A) term, making them unable to work in person on hardware during that term.

References

[1] R. Raskar et al., “Prakash: Lighting Aware Motion Capture using Photosensing Markers and Multiplexed Illuminators” [Online]. Available: https://web.media.mit.edu/~raskar/LumiNetra/Paper/CONFIDENTIALraskarSig07OpticalMocapPaper.pdf. [Accessed: Jun. 29, 2023]

[2] Z. M. Research, “Global 3D Motion Capture System Market Size: Revenue Valued at USD 180 Million in 2022, Expected to Reach USD 600 Million by 2030 at a CAGR of 14.12%,” GlobeNewswire News Room, Apr. 04, 2023. https://www.globenewswire.com/en/news-release/2023/04/04/2640330/0/en/Global-3D-Motion-Capture-System-Market-Size-Revenue-Valued-at-USD-180-Million-in-2022-Expected-to-Reach-USD-600-Million-by-2030-at-a-CAGR-of-14-12.html#:~:text=filingsmedia%20partners-.
[Accessed Jun. 29, 2023]

[3] “How much does a motion capture system cost? - Racket Source,” Racket Source, Apr. 06, 2022. [Online]. Available: https://www.racketsource.com/articles/how-much-does-a-motion-capture-system-cost-cb40dc73. [Accessed: Jun. 29, 2023]

[4] “Dimensions: [mm].” [Online]. Available: https://www.we-online.com/components/products/datasheet/1541201NEA400.pdf [Accessed: Feb 5, 2024]

[5] “VSMA1094250.” [Online]. Available: https://www.vishay.com/docs/80299/vsma1094250.pdf [Accessed: Feb 5, 2024]

[6] “Three-Dimensional Plotting in Matplotlib | Python Data Science Handbook,” jakevdp.github.io. https://jakevdp.github.io/PythonDataScienceHandbook/04.12-three-dimensional-plotting.html [Accessed: Feb 21, 2024]